EE 332 Midterm Examination

February 15, 2002

- 2 Hours
- Open Book
- Two pages plus example code

1. 25 Marks (5 marks each)

   (a) Why is DMA controller access to main memory in most systems given higher priority than CPU access to main memory?

   (b) Some processors provide an instruction that allows the processor to cease fetching and executing instructions until an interrupt occurs. Why is this desirable and useful? Where is such an instruction often used in a full-featured real-time executive?

   (c) Given two RTEMS tasks, one with priority 10 and the other with priority 20, which task has the higher priority?

   (d) Briefly describe the differences between call-by-value and call-by-reference and give an example of each.

   (e) A polled loop system polls a signal every 50 microseconds. Testing the state of the signal and transferring control to a processing routine takes 40 microseconds. What is the minimum response time from the signal becoming active till processing of the signal begins? What is the maximum response time?

2. 25 Marks

   RTEMS task state transitions are an expansion the generic real-time executive task state transitions discussed in class:

   - RTEMS divides the generic 'Dormant' state into 'non-existent' and 'dormant' (created but not yet started) states.
   - RTEMS makes a distinction between tasks which are blocked and tasks which are suspended, and tasks which are both suspended **and** blocked.

   Draw the task state transition diagram for RTEMS tasks. To avoid confusing masses of text and arrows, do not attempt to describe the transitions on the diagram but rather label each transition with a number and provide a table describing the transitions. Each table entry should include:

   - A brief description of the transition.
   - The conditions under which this transition will occur including whether the transition can be caused be the task itself or must be caused by the operating system or some other task.
   - The RTEMS task manager directive associated with the transition, where applicable.

3. 25 Marks
Write the C code to create and start an RTEMS task. The task parameters are:

- Task name: `0xC0FFEE`
- Initial priority: 100
- Initial mode: `RTEMS_DEFAULT_MODE`
- Attributes: `RTEMS_FLOATING_POINT | RTEMS_LOCAL`
- Stack size: 12 kbytes
- Entry point: `subTask`
- Argument: 125

12 × 1024 = 12288

Your code should verify that the RTEMS task manager directives succeed. If the directive fails the routine `fatal_error` should be called. The argument to the `fatal_error` routine should be the status code returned by the failing directive.

4. 25 Marks
Write the `TaskCreate` subroutine for the task dispatcher from assignment 2 (the dispatcher code is also attached to this examination paper). The arguments to the subroutine are the base address of the new task's stack area, passed in the accumulator, and the entry point of the new task, passed in the DPH and DPL special function registers.

Your answer must include a high-level description of the functions to be performed as well as the fully-commented assembly-language code to perform those functions. Use the dispatcher code as an example of the instructions available. Do not panic if you are not completely familiar with 8051 assembly language programming. Marks will not be deducted for small errors in the details of the opcodes or operands as long as the high-level description and comments show your intentions.

END

```
        ORG     000BH
Timer0Handler:      ; Assume timer in mode 2, so no housekeeping needed
        ;
        ; Save context
        ; Don't need to push registers R0-R7 since the PSW we pick up
        ; for the new task will use a different register bank (task 0
        ; uses bank0, task 1 uses bank 1, etc.).
        ;
        push    PSW
        push    ACC
        push    B
        push    DPL
        push    DPH

        ;
        ; Save stack pointer
        ;
        mov     A,#TaskStackPointerArray        ; Beginning of array
        add     A,ActiveTask                    ; Indexed by active task number
        xch     A,R0                            ; Set up pointer
        mov     @R0,SP                          ; Save stack pointer
        xch     A,R0                            ; Restore R0

        ;
        ; Update active task number
        ;
        inc     ActiveTask                      ; Move to next task
        mov     A,ActiveTask                    ; Get new task number
        cjne    A,TaskCount,noWrap              ; Past end of tasks?
        clr     ActiveTask                      ; Yes, reset to task 0
noWrap:
        ;
        ; Switch to new task's stack
        ;
        mov     A,#TaskStackPointerArray        ; Beginning of array
        add     A,ActiveTask                    ; Indexed by new task number
        xch     A,R0                            ; Set up pointer
        mov     SP,@R0                          ; Load stack pointer
        xch     A,R0                            ; Restore R0

        ;
        ; Restore context
        ;
        pop     DPH
        pop     DPL
        pop     B
        pop     ACC
        pop     PSW                     ; Switches to new register bank, too
        reti                            ; Return in new task

;
; Variables
;
        ORG     70H
TaskStackPointerArray:  DS      4       ; Maximum of 4 tasks (1 reg bank/task)
TaskCount:      DS      1               ; Number of tasks
ActiveTask:     DS      1               ; Task number of active task
        END
```